The NuMI Beam Simulation with Flugg

Alexander I. Himmel
California Institute of Technology

April 19, 2010

Contents

1 Introduction

This note describes the newest version of the flux simulation for the NuMI beam line. It introduces two major changes. The previous flux simulation versions[4] used a combination of Fluka and Geant3: Fluka to simulate the target and Geant3 to swim the particles coming off the target through the rest of the beamline geometry. The new simulation also uses a combination of Fluka and Geant, but uses the Flugg[3][2][1] package to simulate a Geant4 geometry in Fluka. Instead of having a two step simulation process, now all of the geometry is handled by Geant and all of the interactions are handled by Fluka. Interactions in the downstream regions are now handled by the same, more trusted, hadronic interaction model as interactions in the target. The largest advantage of this change is in the simulation of helium interactions in the decay pipe. The Geant4 geometry used is also a significant upgrade from the earlier version. It includes more detailed descriptions of many of the beamline components (horns, decay pipe window, hadron absorber, muon monitors) as well as more accurate dimensions based on as-built information where available.

2 How to Run

2.1 Setup and Installation

First, you will need working copies of Geant4,[1] Fluka,[2] and Flugg;[3] instructions for installing all three can be found online. You will need to install the first two before installing Flugg. The code required for g4numi in Flugg can be found in two directories in CVS:

```
numisoft/g4numi
numisoft/g4numi_flugg
```

After checking out the two directories, run `g4numi_flugg/scripts/setup.sh`. This sets up the symlinks from the geometry source files in `g4numi` to `g4numi_flugg`. It also replaces a few source files in the main Flugg source directory, fixing some bugs that prevent the simulation from running smoothly.[4] Be sure to setup the Flugg environment first. This is necessary so the script can put the patched Flugg source files in the right location. After this, compile the executable by running `gmake` in `g4numi_flugg/`.

2.2 Running the Simulation

The process of running Flugg can be somewhat onerous so a script was written that automates the process. You can find it here:

```
g4numi_flugg/scripts/g4numi_fluka.sh
```

In order to simplify running on a batch system, it does not take command line arguments but instead uses environmental variables to configure it. Table 1 lists the required variables and Table 2 lists the optional ones. The script will automatically create a parent directory based on the beam configuration and then a sub directory for each run like this:

```
$DATA/flugg_le<z position>z<current>i_run<run period>/Run<job number>
```

for example,

```
$DATA/flugg_le010z185i_run2/Run109
```

Since the random seed for each run is based off the run number, individual jobs can be rerun and will return the same results. Table 3 shows the files you can expect to find in the job directory of a successful run.

Variable	Example	Description
`RUN`	`109`	The run number for this job. It defines the output names and the random seed.
`EVTS`	`500000`	The number of protons-on-target to simulate.
`PERIOD`	`2`	Run Period (1-4). Periods 3 and 4 have Helium automatically.
`DATA`	`$HOME/data/flugg`	Directory where files are output.

Table 1: Required environmental variables for `g4numi_fluka.sh`.

[1]Geant4: http://geant4.web.cern.ch/
[2]Fluka: http://www.fluka.org/
[3]Flugg: http://www.fluka.org/content/tools/flugg/
[4]This is no longer the default behavior. The patched files are still distributed but due to frequent version clashes they are not automatically copied into the Flugg source tree.

Variable	Default	Description
`CURRENT`	`185`	The nominal horn current in (kA). It is converted to the actual value internally (e.g. `185` $\rightarrow$ 182.1 kA).
`TARGETZ`	`10`	The nominal target z position in -cm. `10` becomes -8.9 cm automatically for runs II and III.
`STEPL`	`1.0`	The maximum step size in cm. Be sure to include the decimal point, for example `export STEPL="1.0"`
`SPECIAL`	-	The contents of special is added to the run type. For example, if `SPECIAL="_sh"`, then the files would output to `flugg_le010z185i_run1_sh/`
`LOWTH`	-	Set to "yes" to remove the 1 GeV tracking threshold
`TARGFILE`	-	Set to "yes" to produce target hadron ntuples in addition to the neutrino files.

Table 2: Optional environmental variables for `g4numi_fluka.sh`.

File	Description
`flugg_le010z185i_run2_109.ntp`	The neutrino ntuple in hbook format.
`flugg_le010z185i_run2_109.root`	The neutrino ntupe in root format.
`target109.ntp`	The target hadron ntuple in hbook form. (Optional)
`target109.root`	The target hadron ntuple in root form. (Optional)
`flugg_le010z185i_run2_109.out`	The standard out from the Fluka parts of the simulation
`flugg_le010z185i_run2_109.err`	The standard error from the Fluka parts of the simulation.
`flugg_le010z185i_run2_109.log`	The standard out and error from the Flugg/Geant parts of the simulation.
`g4numi.inp`	The Fluka input file created by `g4numi_fluka.sh`.
`runConfig.inp`	The geometry configuration file used by the (geant part) of the executable.
`g4numi001_flukaMat.inp`	The Fluka material configuration created by Flugg (see Section 2.3).
`g4numi001_Volumes_index.inp`	File created by Flugg that gives the assosciation between Fluka region numbers and Geant names.

Table 3: Output files found in the directory of a successful job. As an example the files for Run 109 of a le010z185i, Run II job are shown.

2.3 Inside the Scripts

In addition to basic bookkeeping tasks (creating and moving to directories, renaming files, etc.), the script does three major things. First, it creates the Fluka input file (`g4numi.inp`) that configures many aspects of the simulation. Second, it runs the executable using the `rfluka` script provided in the Fluka distribution. Third, it conserves disk space by converting the output from ascii to the two binary file formats we use (hbook and root).

The command used to run the executable is:

```
$FLUPRO/flutil/rfluka -e $FLUGGINSTALL/bin/Linux-g++/mainG4NuMI -N0 -M1 g4numi
```

This tells `rfluka` to use the `mainG4NuMI` executable, run 1 job starting at run number 0 (this is so that the random seed given in the input file is used and fluka doesn't look for an already existing random seed) and to use the `g4numi.inp` input file. `rfluka` creates a temporary directory with the name `fluka_<pid>` in the job directory (`.../Run<job number>`). This is where you will find the log files and partially completed output files for jobs that crash or are still in progress.[5] Some of the filenames will be different from those in a finished run because `rfluka` renames the files when the job completes and `g4numi_fluka.sh` renames some of them again, but it should still be relatively simple to identify the files you care about.

The Fluka simulation does not write directly to binary data, it instead writes one or two ascii files (depending on whether the target ntuple is being produced) which are then converted to the two binary formats with the following scripts/programs in `g4numi_flugg`: `root/fill_flux.C, root/fill_target.C, hbook/ascii2hbook_beam, hbook/ascii2hbook_target`. The former two are root scripts and the latter two are actually small compiled Fortran programs (they should get compiled when you compile the main executable so you should not have to worry about them). The ascii files are only removed when both the hbook and root files are produced successfully.

The input file created by this script configures many aspects of the simulation. The input file starts with a series of cryptic commands (called 'cards' in Fluka jargon), like `USERDUMP` and `MAT-PROP` which are included in order to activate pieces of user written code. There are also cards to turn off particles we do not care about (electrons and photons). This part of the input file is read in from `g4numi_flugg/g4numi.start` and is unlikely to need to be changed by the user.

The next part of the input file assigns materials to regions of the geometry. This part is read in from a file generated automatically by Flugg. This is why, under normal circumstances, Flugg simulations need to be run twice, first to produce this file and second to do the full simulation.[6] For the geometry as it exists in CVS, these material assignment files are generated ahead of time and can be found in `g4numi_flugg/g4numi_flukaMat_<name>.inp`. If you make a change to the geometry that changes the number of regions (adds/removes) or changes the material of a region, you will need to add your own `flukaMat` file and edit `g4numi_fluka.sh` to read that file in instead. The easiest way to get the file would be to run the script as usual and stop the processing by hand after 30 seconds or so (no reason to run it longer since its results will be nonsense with the wrong materials file). You can then find the correct flukaMat.inp file in the `<flugg_run_dir>/Run<process no>/fluka_<pid>` directory.

After the material definitions comes the proton beam configuration. The energy and divergence are always the same, though the size of the beamspot changes for different run periods. The beam starts at -6 m to make sure it starts ahead of the target for all beam configurations. After the beam are a few settings turned on or off by some of the environmental variables described in Table 2. These cards do things like set the step size, turn off multiple scattering, etc. The final entries of the input file set the random seed, set the number of protons to simulate, and call the user code that opens and closes the output files.

In addition to the main `g4numi.inp` file, `g4numi_fluka.sh` also produces a very small configuration file called `runConfig.inp` that the executable reads to determine some parameters that affect how the geometry is constructed (run period, target position, horn current).

3 The Code

3.1 Overview

The Flugg simulation code falls into two broad categories: the main simulation, written in Fluka (Fortran) and the geometry description, written using Geant4 (c++). The basic idea for the interaction between these two parts is this: Fluka performs the actual simulation (performs particle interactions, tracks particle properties, writes output files), but when Fluka goes to make a query

[5] You can identify the crashes since they will have core dumps (`core.<pid>`) if your system allows them.

[6] This is described in more detail at the Flugg website, http://www.fluka.org/content/tools/flugg/node8.html

about the geometry, the Flugg code uses wrappers to pass questions like "Where am I?" and "How far is it to the region boundary?"off to Geant4.

Here, in broad strokes, is how the simulation proceeds from primary proton to output neutrino. For more details see the referenced sections. The simulation begins with a primary proton defined in `g4numi.inp` (Section 2.3). When the primary proton enters the target region, its initial properties are recorded by `FLUSCW` and are used for all the particles which come from this proton. While in the target, the proton presumably interacts, producing potentially many secondaries. As these secondaries leave the target volume, their properties are stored in the `(NUMI)` common block, again by `FLUSCW`. All the later generations of particles following from a particular target parent keep a reference to the target parent's entry in the common block in an `ISPUSR` variable. This way, when we get a neutrino we have the information about the particle that produced it as it left the target. This functionality reproduces the behavior of earlier simulations where the target was simulated separately from the rest of the beamline.

Whenever a secondary is produced, inside the target or out, the `STUPRF` user routine is called. It performs two important functions: it calls the importance weighting function (see Section 3.2) and it keeps track of the particle's parent information (see Section 3.3). When the particles leave the target, they and any secondaries they produce are tracked through the rest of the beamline geometry. The new g4numi geometry that is used by Flugg contains a much more complete and detailed geometry description than the previous simulations, including the detailed structure of the horns and the decay pipe window. It also includes simulations of the hadron and muon monitors. A later note will describe the geometry simulation in greater detail.

Whenever one of these tracked particles decays, the `USDRAW` user routine gets called. This routine loops through all of the created secondaries. For each secondary that is a neutrino, an entry is written to the neutrino ntuple. Note, we do not generally use the neutrino direction chosen randomly by the decay within the simulation but instead use the neutrino's parent information to calculate a weight (or probability) for that neutrino to reach the center of the near or far detectors. This weight is calculated by the `NUWEIGHT` function. The entries in these ntuples are described in detail in Section 4.2.

3.2 Importance Weighting and Thresholds

The basic idea behind importance weighting is that we can save disk space and processing time by not simulating every single one of the lower energy particles but instead only keep a fraction of them, but weight that fraction up so that the weighted flux is unchanged. The idea is to more evenly distribute the statistics and processing time across energies, even though we produce many fewer high energy particles. A parent's weight is passed down to its children, which each may apply their own additional weighting to get their total weight. Every particle has a weight of at least 1, and we set a maximum weight of 100 to avoid having very rare, ultra high weight events.

Importance weighting is performed by the `NUWEIGHT` function which is called by `STUPRF`. The algorithm is as follows. First, the code decides whether or not a particle should be importance weighted. ν's, μ's, and K's are not given any additional weight (though they may carry a weight greater than 1 from their parents). Particles with a total momentum above 30 GeV are also not given any additional weight. For the remaining particles, a candidate weight, W, is calculated as

$$W = 30/|P_{tot}|.$$

We then check to see if this weight would push the particle's total weight ($W_{tot} = W \times W_{parent}$) above 100. If it does, W is set such that $W_{tot} = 100$. We then decide whether to simulate or discard this particle. We do this by choosing a random number R Uniform$(0, 1)$ and if $R > 1/W$ we discard the particle.

There is a general tracking threshold throughout the simulation of 1 GeV. That is, if a particle ever has an energy of below 1 GeV, it is discarded. While this is fine for MINOS, for other off-axis experiments lower energy fluxes may be needed. This threshold can be turned off completely from the run scripts with the `LOWTH` variable (see Table 2 in Section 2.2). Alternatively, a different threshold can be set by editing the `PART-THR` card which is defined in `g4numi_fluka.sh`.

3.3 Particle History Tracking

STUPRF is called whenever a particle is added to the "stack," the collection of particles that have been produced but have not been tracked yet. When STUPRF is called, we have both the parent properties, stored in the TRACKR block, and the child properties, stored in GENSTK block. It is here that the spare tracking variables described in Table 4 are filled. The current position, which is the production point of the secondary, is stored in SPAUSR(1-3). Since each secondary is really just a potential neutrino parent, this point is also referred to as the '(neutrino) parent production point.' The current momentum of the secondary, that is the momentum at its production point, is also stored here in SPAUSR(7-9). The parent production medium is recorded based on the properties of the current region and the parent species is also recorded.

Finally, the SPAUSR(4-6) variables are set based on the parent of the current secondary (or potential neutrino grandparent), i.e. using the TRACKR variables. If the current secondary is a muon, these variables are set to the grandparent's decay momentum (the current momentum in TRACKR). Otherwise, the variables are set to the values in SPAUSR(7-9), that is the production point information of the grandparent. This dichotomy occurs because these same three slots need to be used for two different purposes due to having a limited number of available SPAUSR variables. For muons, the grandparent decay information is needed in order to calculate the effects of muon polarization on the neutrino weight at the detectors. For non-muon parents, the grandparent production point information is needed to fill the tgp* variables in the ntuple in a way equivalent to earlier simulations.[7]

Variable	Description
SPAUSR(1-3)	x, y, z at neutrino parent production (grandparent decay) point
SPAUSR(4-6)	px, py, pz of the neutrino grandparent at the decay point (muon parents) or production point (hadron parents)
SPAUSR(7-9)	px, py, pz at the neutrino parent production point
SPAUSR(10)	Particle importance weight
ISPUSR(1)	Index in (NUMI) of the target parent
ISPUSR(2)	Neutrino grandparent species
ISPUSR(3)	Neutrino great-grandparent species
ISPUSR(4)	Flag to kill particles. Set to 1 to keep a paritcle and 2 to kill it.
ISPUSR(5)	1 if SPAUSR(4-6) refers to a muon parent
ISPUSR(6)	Parent production mode flag (1 = inelastic, 2 = decay, 0 = other)
ISPUSR(7)	Parent production medium

Table 4: Describes how all of the Fluka spare tracking variables ISPUSR[] and SPAUSR[] are used.

3.4 Fluka Source Files

for/(NUMI)

A common block to store target parent information.

for/fluscw.f

The FLUSCW routine is called in a number of circumstances, but the one used is that it is called whenever a boundary is crossed. In particular, it is used to catch when particles enter or leave the target region. Whenever a proton enters the target region, its information is stored and used for all target secondaries coming from that proton. Whenever a particle leaves the target region, its properties are stored in an entry in the (NUMI) common block (the block is reset at each new proton). The entry number is stored in an ISPUSR variable for that particle which is then passed on to all later generations coming from that particle. That way, when a neutrino is created we

[7] For the production run of files, a bug caused only the decay point to be recorded, not the production point. This means that the mupar* variables and the tgp* variables all refer to the decay point for all particles.

can access the target parent information. If the target hadron ntuple is being written out, that happens here as well (the hadron ntuple is described in Table 8 in Section 4.2). It contains all the target secondaries that are added to the (NUMI) common block.

`for/gcode.f`
A routine that converts Fluka particle codes to Geant particle codes. It is called while the neutrino ntuple is being written out in `MGDRAW`.

`for/impwgt.f`
Implements the importance weighting scheme. See Section 3.2 for more details.

`for/magfld.f`
Passes the Fluka magnetic field call off to the Geant4 wrapper so that the Geant4 field is used. This should not require modification by the user.

`for/mcode.f`
A routine that converts Fluka material codes to the Gnumi material codes. If a given fluka material does not have an equivalent in the Gnumi scheme, the function returns the original fluka material multiplied by -1. It is called while the neutrino ntuple is being written out in `MGDRAW`.

`for/mgdraw.f`
`MGDRAW` has numerous subroutines. The one used in this simulation is `USDRAW`. It is called after every particle interaction. The process of interest is `ICODE = 102` which refers to decay. The neutrino ntuple is written out here (the neutrino ntuple is described in Table 6 and Table 7 in Section 4.2). Whenever there is a decay, the function loops through the secondaries. For each neutrino found, an entry is written to the ntuple. Note, when the ntuple is being written the `TRACKR` variables still refer to the neutrino *parent* at its decay point, not the neutrino.

`for/nuweight.f`
The `NUWEIGHT` routine defined in this file calculates the near and far detector (or any other position) weights. It takes polarization into account for muons.

`for/stuprf.f`
The `STUPRF` routine is called whenever a new particle is added to the stack, i.e. when we get new particles from a decay or an inelastic interaction. This routine is responsible for the history tracking or "latching." The particles own production point information is stored (position and momentum) along with the momentum of the particle's parent at either production (muons) or decay (hadrons)(at the decay point in production files – see footnote on page 6). The variables are described in Table 4.

`for/usrini.f`
This user routine is called when the executable first begins to run. It opens the needed output ascii files.

`for/usrmed.f`
`USRMED` is used normally to apply custom material properties. Here it is used to allow importance weighting to kill particles. If another part of the code (e.g. `IMPWGT`) decides to kill a particle, it sets `ISPUSR(4) = 2` which tells `USRMED` to zero its weight, stopping it from being further simulated.

`for/usrout.f`
This user routine is called at the end of the Fluka run. It closes the opened output files.

3.5 Geant Source Files

`include/NumiDataInput.hh, src/NumiDataInput.cc`
The `NumiDataInput` class creates a singleton object that holds all of the configurable parameters of the beamline geometry. It reproduces the functionality of the input files used in the Geant3-based Gnumi. It is this class that reads in the `runConfig.inp` file described in Section 2.3. It is

used by all the other geometry files.

`include/NumiDetectorConstruction.hh, src/NumiDetectorConstruction.cc`
This is the starting point for constructing the NuMI geometry and contains the method actually called by Flugg (`NumiDetectorConstruction::Construct()`). NumiDetectorConstruction.cc is itself mostly just a jumping off point that calls separate methods contained in separate files that produce the rest of the geometry. These files are listed below and their names are self-explanatory. Since they are all technically just a part of the `NumiDetectorConstruction` class they do not have their own associated header files.

```
src/NumiBaffle.cc
src/NumiDecayPipe.cc
src/NumiHadronAbsorber.cc
src/NumiHorn1.cc
src/NumiHorn2.cc
src/NumiMaterials.cc
src/NumiSecMonitors.cc
src/NumiTarget.cc
src/NumiTargetHall.cc
```

`include/NumiHornSpiderSupport.hh, src/NumiHornSpiderSupport.cc`
A specialized object used for creating the horn support structures.

`include/NumiMagneticField.hh, src/NumiMagneticField.cc`
Actually contains three classes that describe the magnetic field in the inner conductor, in the outer conductor, and in the bulk of the horn volume.

Note: These are not all of the source files in g4numi but rather just the subset that are used by Flugg.

4 The Output Ntuples

4.1 Available Files

The Flugg flux files can be found at Fermilab. They are available in `/minos/data/flux/flugg/` with subdirectories for each run configuration. The available run configurations are described in Table 5.

Beam	No. Files	Target	Target Z (cm)	Current (kA)	Decay Pipe
le010z000i_run1	317	NT01	-10	0	Vacuum
le010z000i_run2	449	NT02	-8.9	0	Vacuum
le010z000i_run3	440	NT02	-8.9	0	Helium
le010z170i_run1	441	NT01	-10	167.3	Vacuum
le010z185i_run1	441	NT01	-10	182.1	Vacuum
le010z185i_run2	431	NT02	-8.9	182.1	Vacuum
le010z185i_run3	442	NT02	-8.9	182.1	Helium
le010z185i_run4	442	NT03	-10	-182.1	Helium
le010z200i_run1	445	NT01	-10	196.9	Vacuum
le100z200i_run1	446	NT01	-100	196.9	Vacuum
le150z200i_run2	445	NT02	-150	196.9	Vacuum
le250z200i_run1	445	NT01	-250	196.9	Vacuum
le250z200i_run2	448	NT02	-250	196.9	Vacuum

Table 5: Available files at FNAL. You can access them at `/minos/data/flux/flugg/`.

4.2 File Contents

Variable	Description
`run`	Run number (not used)
`evtno`	Event number (proton on target)
`Ndxdz` `Ndydz`	Neutrino direction slopes for a random decay
`Npz`	Neutrino momentum (GeV/c) along the z-axis (beam axis)
`Nenergy`	Neutrino energy (GeV) for a random decay
`NdxdzNea` `NdydzNea`	Direction slopes for a neutrino forced towards the center of the near detector
`NenergyN`	Energy for a neutrino forced towards the center of the near detector
`NWtNear`	Weight for a neutrino forced towards the center of the near detector
`NdxdzFar` `NdydzFar`	Direction slopes for a neutrino forced towards the center of the far detector
`NenergyF`	Neutrino energy (GeV) for a decay forced to the center of the far detector
`NWtFar`	Neutrino weight for a decay forced to the center of the far detector
`Norig`	Not used in flux file
`Ndecay`	Decay process that produced the neutrino, see Table 11
`Ntype`	Neutrino flavor. $\nu_\mu = 56, \bar{\nu}_\mu = 55, \nu_e = 53, \bar{\nu}_e = 52$

`Vx`
`Vy`
`Vz`
	Neutrino production vertex (cm)

`pdPx`
`pdPy`
`pdPz`
	Momentum (GeV/c) of the neutrino parent at the neutrino production vertex (parent decay point)

`ppdxdz` `ppdydz`	Direction of the neutrino parent at its production point (which may be in the target)
`pppz`	z momentum (GeV/c) of the neutrino parent at its production point
`ppenergy`	Energy (GeV) of the neutrino parent at its production point
`ppmedium`	Code for the material the neutrino parent was produced in (see Table 10)
`ptype`	Neutrino parent species (GEANT codes)

`ppvx`
`ppvy`
`ppvz`
	Production vertex (cm) of the neutrino parent

`muparpx`
`muparpy`
`muparpz`
	Momentum (GeV/c) of the neutrino grandparent at the grandparent decay point (muons) or grandparent production point (hadrons) (at the decay point in production files – see footnote on page 6)

`mupare`	Energy (GeV) of the neutrino grandparent, as above
`Necm`	Neutrino energy (GeV) in the center-of-mass frame
`Nimpwt`	Importance weight of the neutrino

`xpoint`
`ypoint`
`zpoint`
	Debugging hook – unused

`tvx`
`tvy`
`tvz`
	Position (cm) of the neutrino ancestor as it exits target (possibly, but not necessarily, the direct neutrino parent)

Table 6: The entries stored in the neutrino ntuple files. There is one entry for every neutrino produced.

Variable	Description
`tpx` `tpy` `tpz`	Momentum (GeV/c) of the ancestor as it exits target
`tptype`	Species of the ancestor exiting the target (GEANT codes)
`tgen`	Neutrino parent generation in cascade. 1 = primary proton, 2 = particles produced by proton interaction, 3 = particles from
`tgptype`	Species of the parent of the particle exiting the target (GEANT codes)
`tgppx` `tgppy` `tgppz`	Momentum (GeV/c) of the parent of the particle exiting the target at the parent production point (at the decay point in production files – see footnote on page 6)
`tprivx` `tprivy` `tprivz`	Primary particle interaction vertex (not used)
`beamx` `beamy` `beamz`	Primary proton origin (cm)
`beampx` `beampy` `beampz`	Primary proton momentum (GeV/c)

Table 7: The entries stored in the neutrino ntuple files. There is one entry for every neutrino produced.

Variable	Description
`x` `y` `z`	Position (cm) of the particle as it exits target
`px` `py` `pz`	Momentum (GeV/c) of the parent as it exits target
`type`	Species of the particle leaving the target (Fluka codes)
`weight`	Weight
`gener`	Generation
`momtype`	Species of the parent of the particle exiting the target (Fluka codes)
`mompx` `mompy` `mompz`	Momentum (GeV/c) of the parent of the particle exiting the target at the parent decay point
`protvx` `protvy` `protvz`	Primary particle interaction vertex (not used)
`protx` `proty` `protz`	Primary proton origin (cm)
`protpx` `protpy` `protpz`	Primary proton momentum (GeV/c)
`event`	Event number

Table 8: The entries stored in the hadron ntuple files. There is one entry for each hadron that exits the target volume.

4.3 Useful Codes

Particle	**Fluka** Hadron File	**Geant** Flux File	**PDG** SNTP
γ	7	1	22
e^+	4	2	−11
e^-	3	3	11
μ^+	10	5	−13
μ^-	11	6	13
π^0	23	7	111
π^+	13	8	211
π^-	14	9	−211
K_L^0	12	10	130
K^0	24	10/16	311
$\bar{K}^0$	25	10/16	−311
K^+	15	11	321
K^-	16	12	−321
n	8	13	2112
p	1	14	2212
$\bar{p}$	2	15	−2212
K_S^0	19	16	310
Λ	17	18	3122
Σ^+	21	19	3222
Σ^0	22	20	3212
Σ^-	20	21	3112
Ξ^0	34	22	3322
Ξ^-	36	23	3312
Ω^-	38	24	3334
$\bar{n}$	9	25	−2112
$\bar{\Lambda}$	18	26	−3122
$\bar{\Sigma}^-$	31	27	−3222
$\bar{\Sigma}^0$	32	28	−3212
$\bar{\Sigma}^+$	33	29	−3112
$\bar{\Xi}^0$	35	30	−3322
Ξ^+	37	31	−3312
Ω^+	39	32	−3334
τ^+	41	33	−15
τ^-	42	34	15
$\bar{\nu}_e$	6	52	−12
ν_e	5	53	12
$\bar{\nu}_\mu$	28	55	−14
ν_μ	27	56	14
$\bar{\nu}_\tau$	44	–	−16
ν_τ	43	–	16

Table 9: The particle codes across the three schemes used in MINOS.

Code	**Material**
5	Beryllium
6	Carbon
9	Aluminum
10	Iron
11	Slab Steel
12	Blu Steel
15	Air
16	Vacuum
17	Concrete
18	Target
19	Rebar Concrete
20	Shotcrete
21	Variable Density Aluminum
22	Variable Density Steel
23	1018 Steel
24	A500 Steel
25	Water
26	M1018 Steel
28	Decay Pipe Vacuum
31	CT852

Table 10: The material codes as defined by Gnumi and used in the fluxfiles, old and current.

Ndecay	**Process**
1	$K_L^0 \to \nu_e + \pi^- + e^+$
2	$K_L^0 \to \bar{\nu}_e + \pi^+ + e^-$
3	$K_L^0 \to \nu_\mu + \pi^- + \mu^+$
4	$K_L^0 \to \bar{\nu}_\mu + \pi^+ + \mu^-$
5	$K^+ \to \nu_\mu + \mu^+$
6	$K^+ \to \nu_e + \pi^0 + e^+$
7	$K^+ \to \nu_\mu + \pi^0 + \mu^+$
8	$K^- \to \bar{\nu}_\mu + \mu^-$
9	$K^- \to \bar{\nu}_e + \pi^0 + e^-$
10	$K^- \to \bar{\nu}_\mu + \pi^0 + \mu^-$
11	$\mu^+ \to \bar{\nu}_\mu + \nu_e + e^+$
12	$\mu^- \to \bar{\nu}_\mu + \nu_e + e^-$
13	$\pi^+ \to \nu_\mu + \mu^+$
14	$\pi^- \to \bar{\nu}_\mu + \mu^-$
999	Other

Table 11: The decay codes stored in `Ndecay`.

5 Validation

The validation section consists of comparisons of the new Flugg flux (v20) to the previous Gnumi flux (v19). The first subsection discusses known changes and their effects on the flux. In this section, if a plots has more than one histogram, the red refers to Flugg and the black refers to Gnumi. If a plot has only one red histogram, then it is the ratio of Flugg to Gnumi. The second subsection includes a more comprehensive set of plots looking at the near detector energy spectrum and far-over-near ratio for the folowing: $\nu_\mu$in every beam configuration and normal low energy (le010z185i) for every neutrino species ($\nu_\mu, \bar{\nu}_\mu, \nu_e, \bar{\nu}_e$) and every major parent species ($\pi^\pm, K^\pm, K^0_L, \mu^\pm$).

5.1 Notable Changes

5.1.1 The Focusing Peak

There are, in fact, two observable changes in the focusing peak: a decrease in the height of the peak and a shift towards higher energy. We can separate out these two effects, and thus get a hint of their cause, by looking at where the neutrino was produced. Figure 1 shows the focusing peak for neutrinos with vertices in the chase and Figure 2 shows the peak for neutrinos with vertices in the decay pipe. We see in the former figure that there is just a reduction for a certain energy range without any shift in the spectrum. What is happening is that some parents that would decay to neutrinos are instead being absorbed by the additional horn material that has been added in this simulation. It happens only for a particular range of energy because not all initial parent momenta

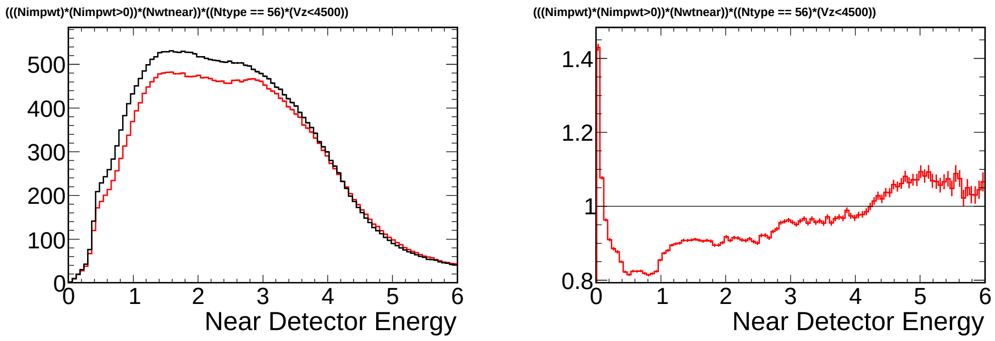

Figure 1: The focusing peak for neutrinos with vertices in the chase ($V_z < 45$ m). We see an energy-dependent decrease in the peak height without any horizontal shift. The integrated number of these events changes by about 7%. Neutrinos with vertices in the chase makes up about 18% of the focusing peak.

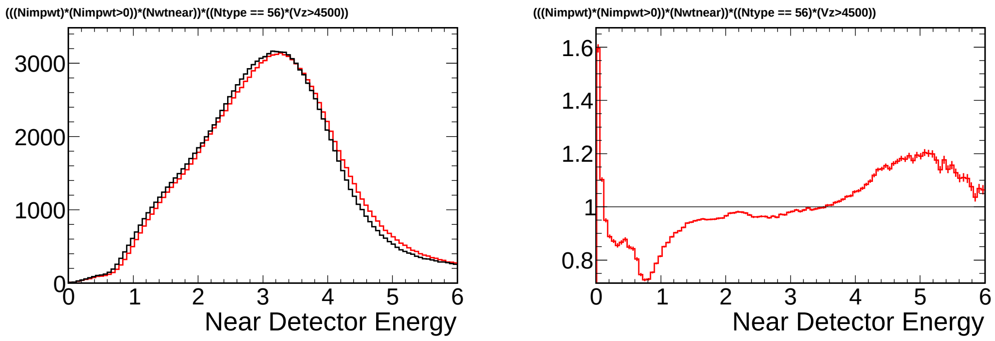

Figure 2: The focusing peak for neutrinos with vertices in the decay pipe ($V_z > 45$ m). We see a horizontal shift the peak without a change in its width or eight. The mean shifts by 2.3% while the RMS and total integral change by 0.6% and 0.1% respectively.

will have their paths focused to pass through the extra material. This hypothesis has been confirmed with a special simulation run without the extra horn material – the peak decrease is not present in this special run.

5.1.2 Helium and Other Downstream Production

Production due to interactions outside the target was one of the primary motivators for developing the Flugg fluxes, since this is where the previous version of the simulation had difficulty. Specifically, the problem was that there was too much high-x_f production from interactions in the helium – much more than was consistent with the data. This was because downstream production was handled by the obsolete Gfluka package, rather than a modern Fluka version as was used in the target. Now, with a consistent hadron production model, the new Flugg fluxes do, indeed, show a decrease in downstream production, especially at higher energies. It is seen for both for ν_μ's (Figure 3) and $\bar{\nu}_\mu$'s (Figure 4). We also see the benefit of this decrease in better modeling of the change in the flux do to helium. This can be seen, both for ν_μ's and $\bar{\nu}_\mu$'s in Figure 5.

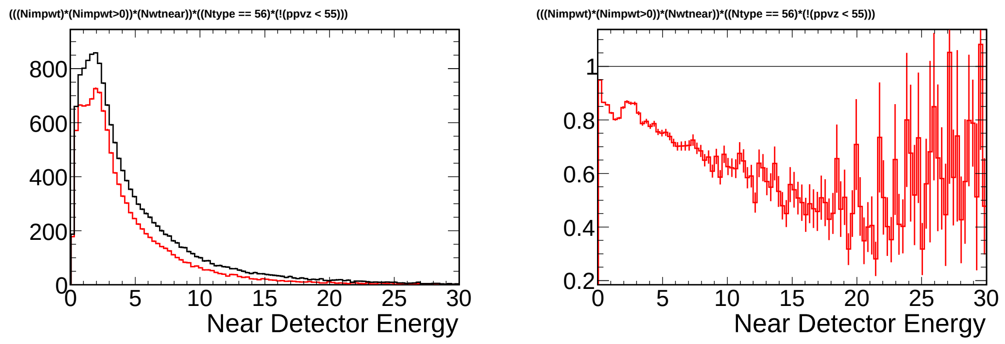

Figure 3: The near detector low energy spectrum for ν_μ's whose parents were produced outside the target. We see a significant decrease (24%), especially towards higher energies.

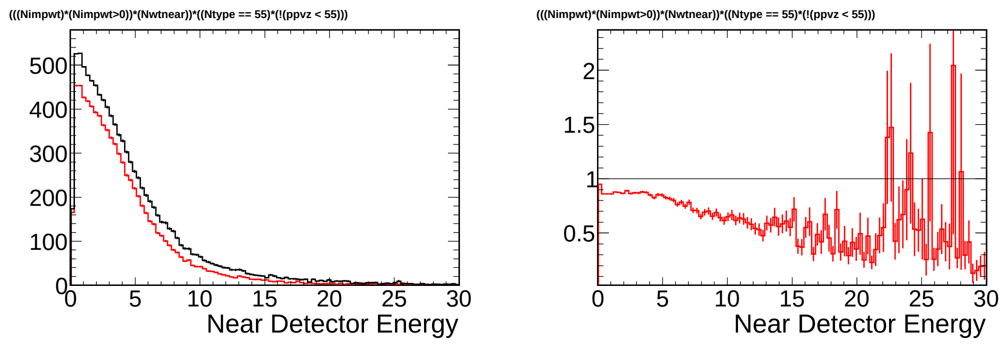

Figure 4: The near detector low energy spectrum for $\bar{\nu}_\mu$'s whose parents were produced outside the target. As with the ν_μ's, we see a significant decrease (20%).

5.1.3 The Near Detector ν_e Flux

Perhaps the most significant unexpected change in the switch to the new flux was a large (14%) increase the ν_e flux at the near detector, as seen in Figure 6. It turns out the change is more generally for all neutrinos from muon decay (16% increase). The other ν_e component, coming from kaon decays, shows an increase of about 3%, both near and far.[8] The muon increase is primarily in

[8]Which is actually quite small compared to the 20% increase in low-energy kaon production due to Flugg's better handling of low energy interactions compared to Gnumi.

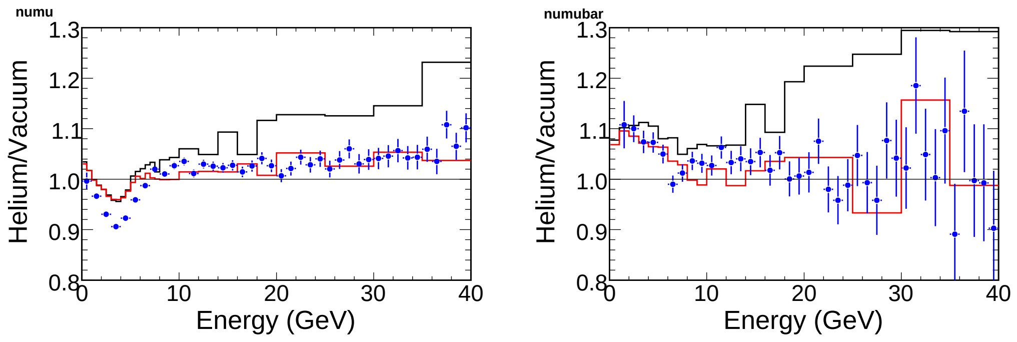

Figure 5: The helium to vacuum flux ratio shown for Flugg in red, Gnumi in black, and the data (Run III over Run II) in blue. The left shows the ratios for ν_μ's and the right shows the ratios for $\bar{\nu}_\mu$'s. It is clear that at higher energies, the Flugg flues are in much better agreement with the data. The discrepancy at low energies in the $\nu_\mu$plot is due to the degradation of the target, not the helium modeling.

the near detector: the far detector and the non-detector weighted fluxes show more modest changes (8% increase and 0.1% increase, respectively).

This behavior is suggestive – the fact that the increase is only apparent for detector-weighted spectra, and is larger for the near detector, points us towards the immediate cause: a change in the V_z (neutrino z vertex) distribution. Figure 8 shows the near detector-weighted V_z distribution. We can see that the increase in near detector flux comes entirely at high-V_z. Figure 9 shows the same distribution but without weighting to the near detector. We see that what appeared to be an increase at high-V_z is actually just a shift from lower V_z to higher V_z without a significant change in overall amount. The change in the overall near detector flux comes from the fact that neutrinos produced towards the end of the decay pipe have a higher probability of hitting the near detector (and thus a higher near detector weight) than those produced at the beginning of the decay pipe. This change in the V_z distribution is not limited to muons, as can be seen in Figures 10-11; however, the affect on ν_μ's is not as pronounced since the pions tend to decay earlier in the decay pipe, making the shift towards higher V_z less pronounced.

The question now becomes why is there a change in the V_z distribution? Unfortunately, this is a difficult question to answer. However, we do have some strong circumstantial evidence that the cause is a change in horn focusing. First, a note about the changes in the Flugg horns. The extra horn material added to Flugg is at the far end in z and at high r. This means that there is an additional focused component that is going straighter but with a wider divergence. This additional 'straightness' might be the cause of the change in the V_z distribution. We also see some evidence of the wider divergence. Figure 12 shows a variable named 'divx' that is calculated by taking the neutrino parent at its decay point and using its final momentum to project it back to what its x coordinate would have been at $z = 45$ m. Both of these distributions do, indeed, show that Flugg has a wider distribution. However, because of the way the variable is constructed, particles that decay closer to the end of the decay pipe will tend to have larger values since they have to project farther back and thus this variable does not necessarily tell us that the raw flux leaving the horns is wider. It gives us the correlation, but it cannot demonstrate causation.

A second piece of circumstantial evidence comes from looking at the relative muon flux, separated by charge, in the different beam configurations. You can see this plotted in Figure 13 and Figure 14. For μ^+'s, as the focused energy increases the near detector excess decreases until there is actually a small deficit in the pseudo high-energy configuration. For the μ^-'s, on the other hand, there is no statistically-significant excess in any focused beam configuration.[9] The fact that changing the horn focusing does affect the flux of μ^+'s but not the flux of μ^-'s strongly suggests (but cannot *prove*) that the horn focusing is the cause of the changes between Flugg and Gnumi. It also cannot tell us

[9]However, there is a significant excess in the horn-off configurations that is not, at this time, understood

what it is about the focusing that makes the difference.

We are limited by not having any information about the immediate muon parent and also by only having information about particles that go on to produce neutrinos. In order to do conclusively prove this hypothesis, information beyond what is stored in the flux files is required, both in Flugg and in Gnumi. This includes more information about the muon parent or a dump of all particles that pass through a plane in the simulation, regardless of whether or not they decay to neutrinos.

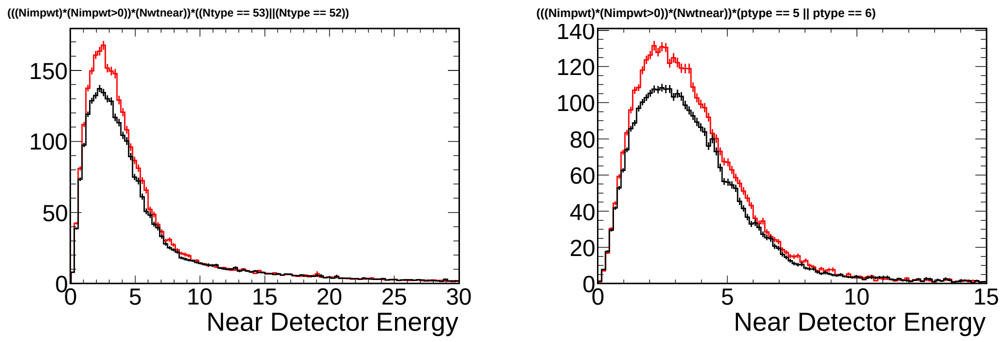

Figure 6: At left is the near detector ν_e and $\bar{\nu}_e$ energy spectrum. It shows a 14% overall increase compared to Gnumi, concentrated at lower energies. At right is the near detector energy spectrum for all neutrino types from $\mu^{\pm}$ decay. Here there is a 16% increase which is relatively consistent for all energies.

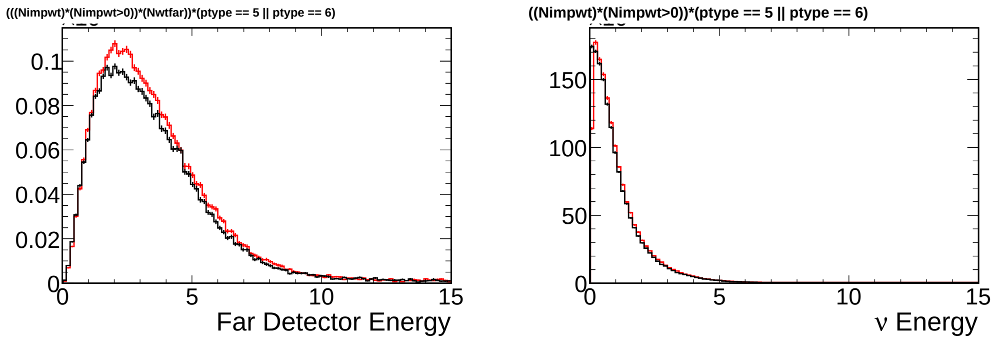

Figure 7: Both figures show the energy spectrum for neutrinos from $\mu^{\pm}$ decay. At left is the far detector spectrum, where Flugg shows an 8% increase relative to gnumi. At right is the non-detector (importance-only) weighted muon flux. Here the total number of muons produced is identical to within 0.1%.

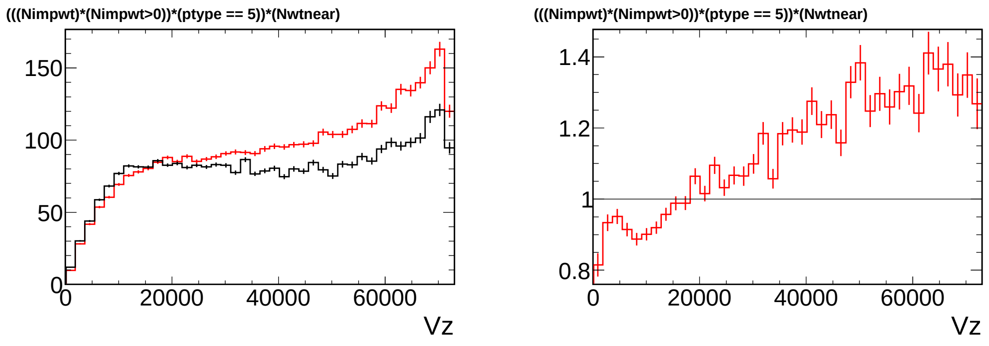

Figure 8: This figure shows the near detector weighted V_z distribution for neutrinos from muon decay. We can see that the 16% increase in flux is concentrated at higher V_z.

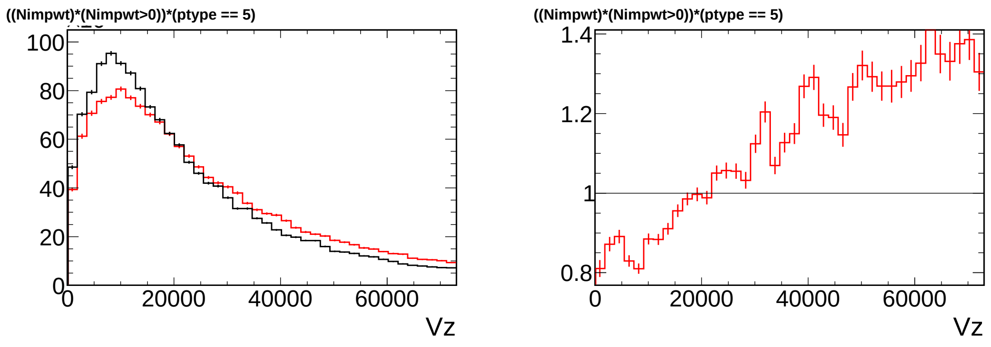

Figure 9: This figure shows the importance (non-detector) weighted V_z distribution for neutrinos from muon decay. We can see that the increase shown in Figure 8 is actually a shift from lower to higher V_z with little change in overall flux when not detector weighted. The increased overall flux comes because neutrinos produced at higher V_z (closer to the near detector) have higher near detector weights (or equivalently, are more likely to hit the near detector).

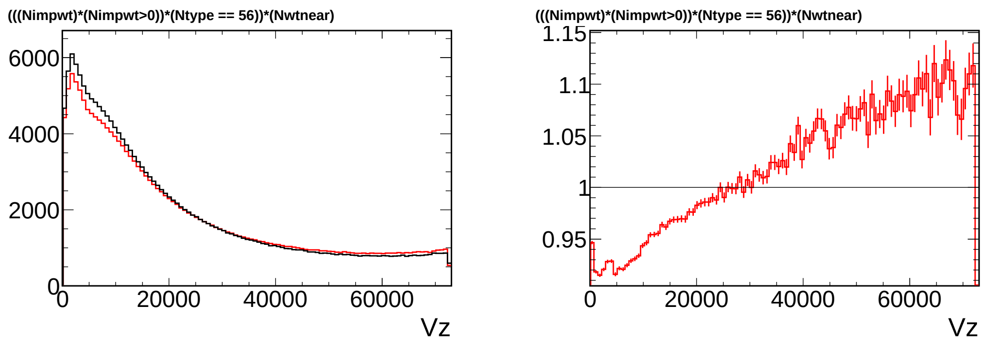

Figure 10: This figure shows the near detector weighted V_z distribution for ν_μ's. While the shift from low to high is still present, the change in overall number of events is much smaller because the distribution is already shifted heavily towards the front of the decay pipe.

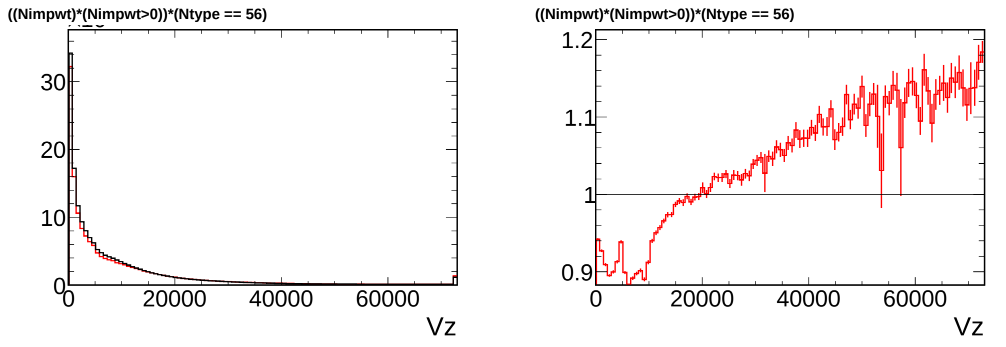

Figure 11: This figure shows the importance (non-detector) weighted V_z distribution for ν_μ's.

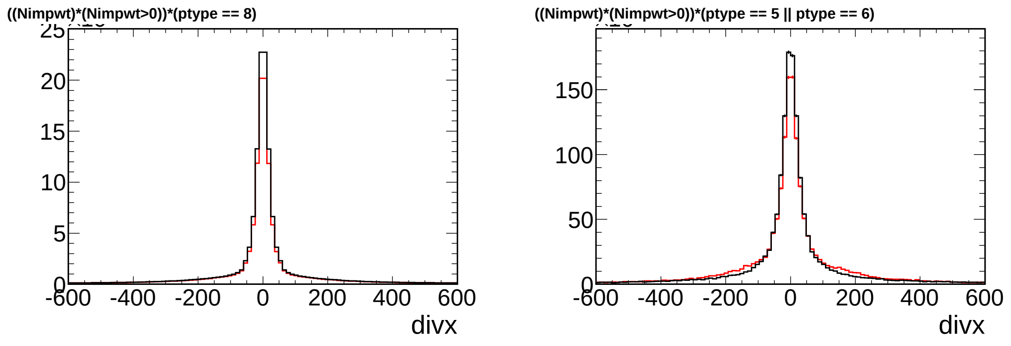

Figure 12: Both of these plots show the *divx* variable – this is the projected x-position the neutrino parent would have passed through at $z = 45$ m based on its final position and momentum. At left is the distribution for pion parents and at right is the distribution for muons parents. The pion RMS is about 3% larger and the muon RMS is about 8% larger. Unfortunately, because the flux files only contain parents that produce neutrinos, these effects may be due to the correlation between this variable and the V_z distribution and cannot demonstrate that Flugg has a wider flux leaving the horns.

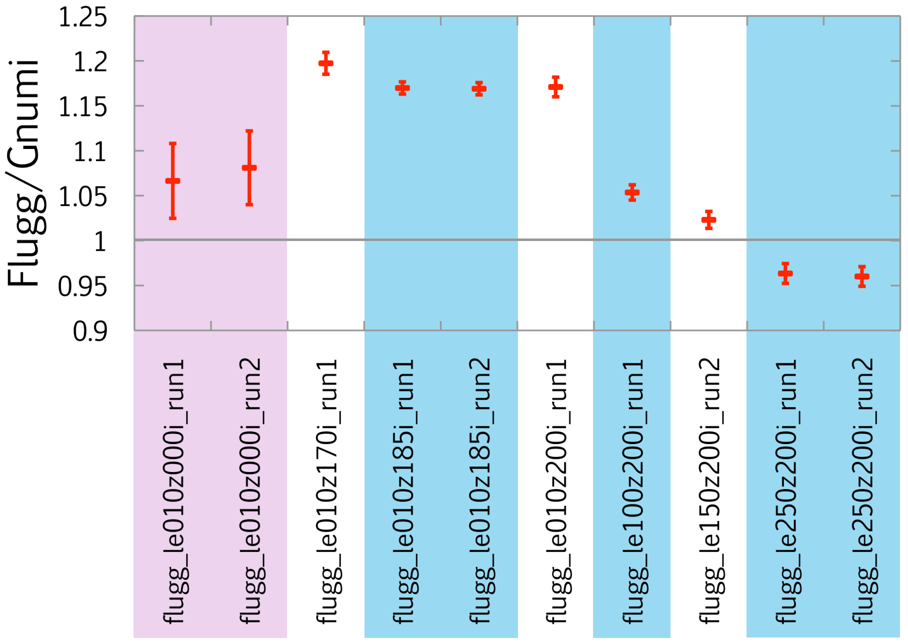

Figure 13: The ratio of integrated μ^+ near detector fluxes from Flugg and Gnumi in a range of beam configurations. The focused energy increases from left to right.

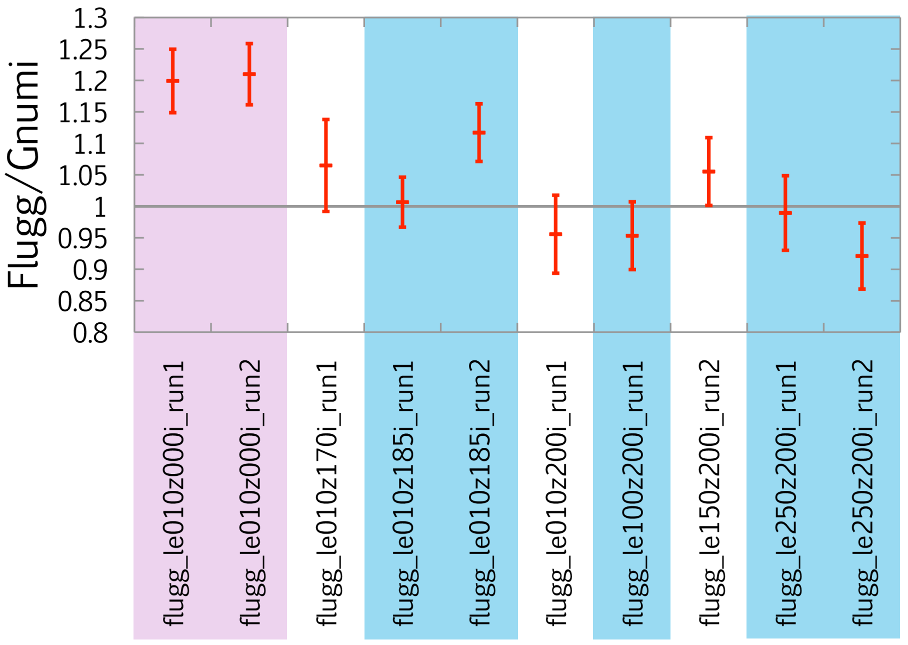

Figure 14: The ratio of integrated μ^- near detector fluxes from Flugg and Gnumi in a range of beam configurations. The focused energy increases from left to right.

5.1.4 Low Angle Scatters

The divergence plots developed to study the ν_e flux show another interesting effect. While the effect pertains to too small a number of events to explain significant spectral differences, it is representative of the types of underlying differences between the behavior two simulations (this one just happened to get tracked down). Using the *divr* variable (the extrapolated radial position at $z = 45$ m, we find that about 5% of the events with parents produced in the target and neutrino vertices in the decay pipe have 'impossible' values. That is, they have values of $divr > 100$ cm, which is the radius of the decay pipe walls. If we plot the *divx* and *divy* for parents that decay in the decay pipe, this class of events can be seen as a halo surrounding the decay pipe entrance (see Figure 15). Since these particles were produced in the target, they must have passed through the central window. Since the halo begins at the radius of the decay pipe, these events seem to be from glancing scatters off the walls. What makes these events interesting is that there are twice as many of them in Flugg and they tend to produce neutrinos farther down in the decay pipe, indicating perhaps smaller scattering angles.

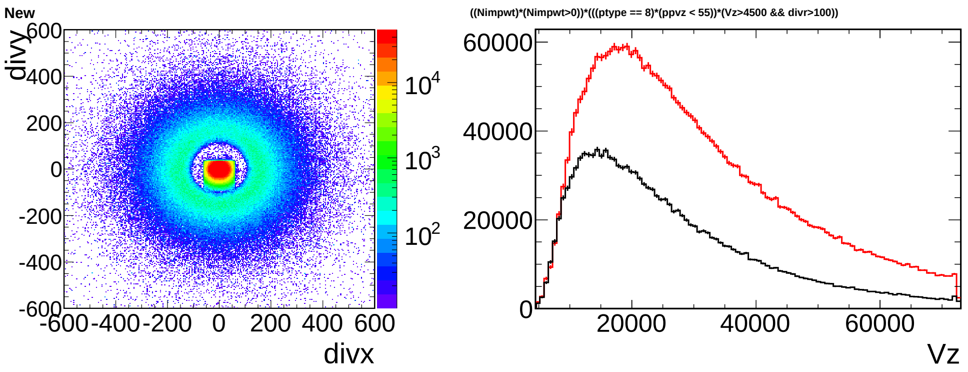

Figure 15: At left the *divy* vs. *divx* positions of parents produced in the target that decay in the decay pipe is shown. The rectangle in the center is the space in which the flux can pass into the decay pipe, defined by the chase shielding. The halo surrounding this window must come from particles deflecting off the decay pipe walls since the halo begins at $r = 100$ cm, which is the radius of the decay pipe. At right are the V_z distributions for events that appear to scatter off the walls of the decay pipe. Flugg has twice as many events and they are shifted to larger V_z values.

5.1.5 Multiple Scattering in the Chase

Another Flugg-Gnumi difference that was found was in the amount of multiple-scattering in the air in the Chase. Specifically, there appears to be none in Gnumi and there is an appreciable amount in Flugg. The effect was discovered looking at a special run with mono-energetic pions traveling directly down the center of the horns. By looking at the neutrino production vertices, we can map out the path taken by these pions. So, by looking at V_x at a slice in z, we can compare the amount of scattering we see with the amount that is calculated for pions traveling through air. This is shown for $z = 45$ m in Figure 16. The calculated 1σ width due to multiple is 1.35 cm and is marked with dashed lines on the plot. You can see that this agrees well with the distribution produced by Flugg.

Further investigation revealed that the Gnumi simulation was originally written with vacuum in the chase. When the simulation was updated to have an air filled chase, the step size in the region was never updated accordingly. So, in Gnumi these particles were taking steps so large that they had no chance for scattering to occur. Again, this difference is too small to affect the flux in a significant way; however, its resolution gives confidence in the Flugg simulation.

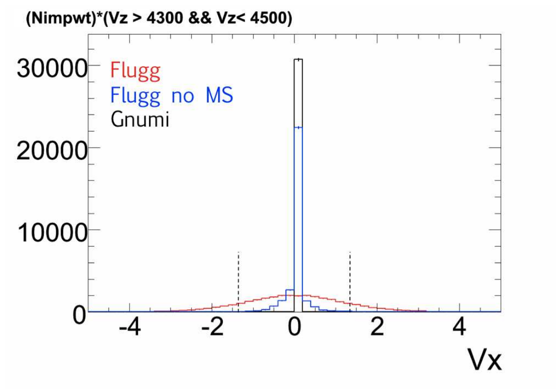

Figure 16: This figure comes from a special run with mono-energetic, neck-to-neck pions. We use the neutrino vertices to show the path of these pions. The figure shows V_x distribution at a slice at $z = 45$ m. You can see that while Gnumi (in black) shows no evidence of scattering, the Flugg distribution matches the calculated 1σ scattering width (1.35 cm - marked with dashed lines) quite well.

5.2 Comprehensive Plots

In the following plots, the color scheme is as follows. Black always refers to gnumi and the colors refer to Flugg in various run periods. The colors are chronological, so red will be the earliest run period, followed by blue, followed by green. You can use Table 5 on page 8 to figure out exactly which run period corresponds to a particular curve in a plot.

5.2.1 Beam Configurations

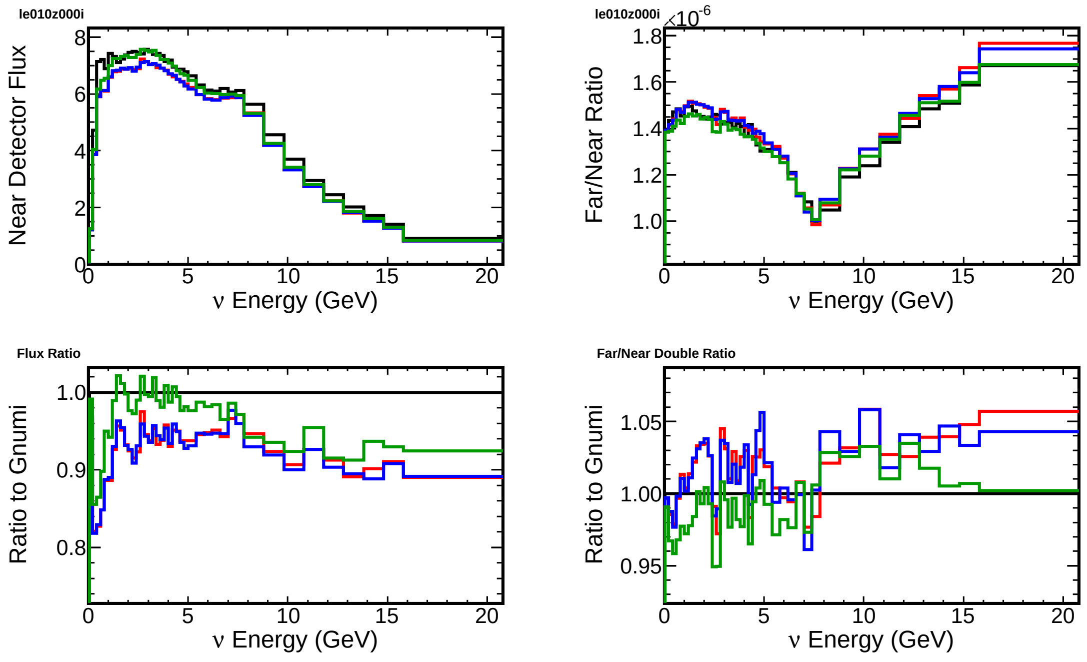

Figure 17: le010z000i, ν_μ

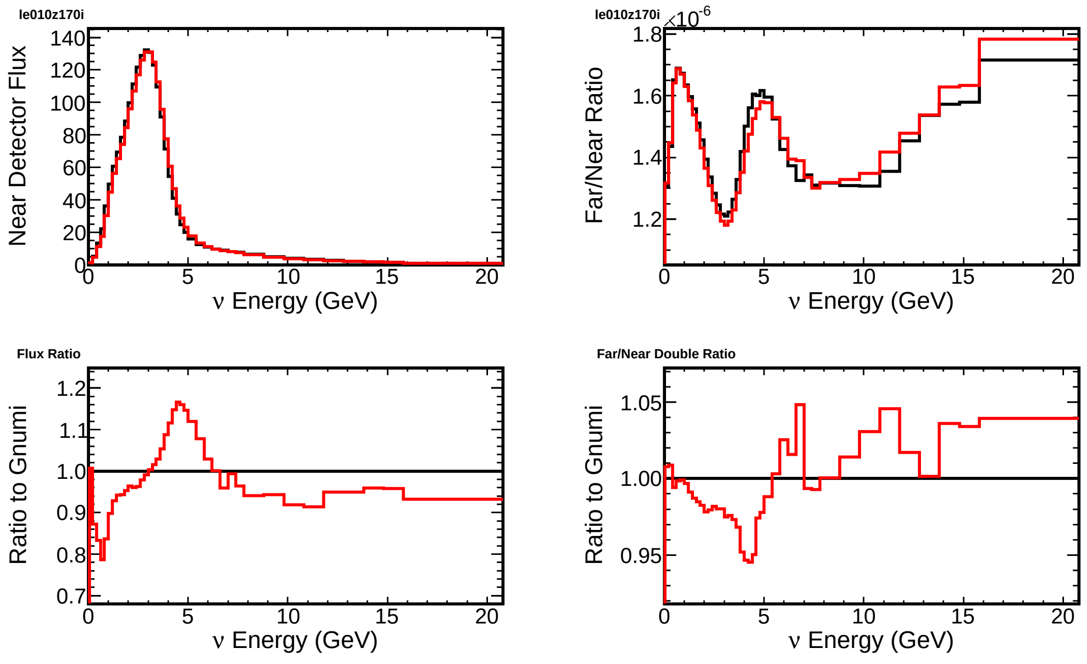

Figure 18: le010z170i, ν_μ

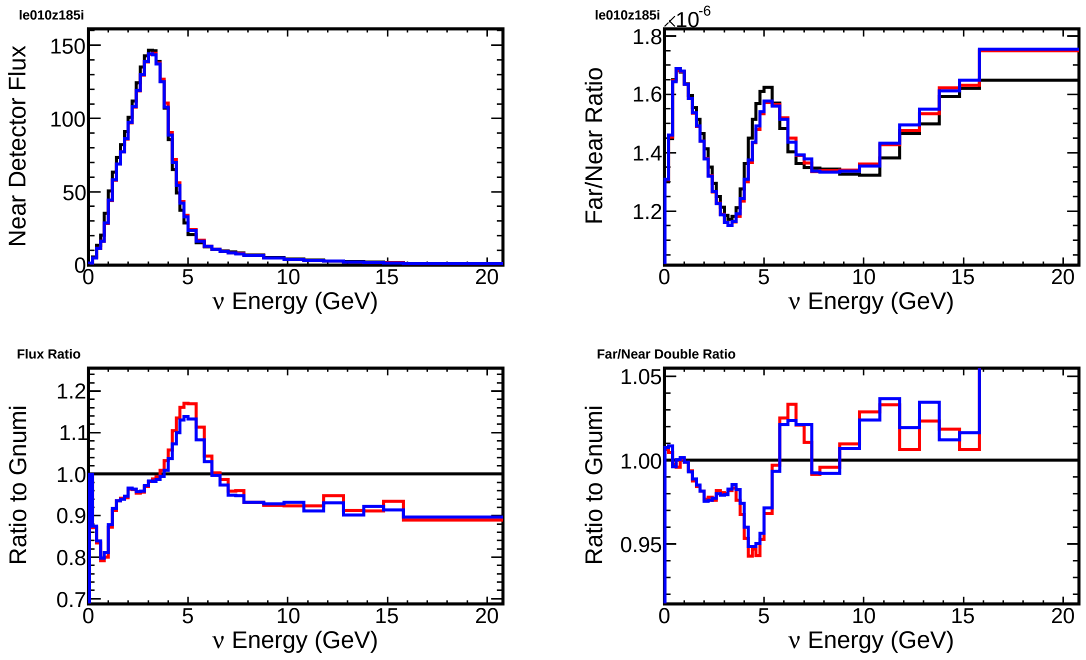

Figure 19: le010z185i, ν_μ

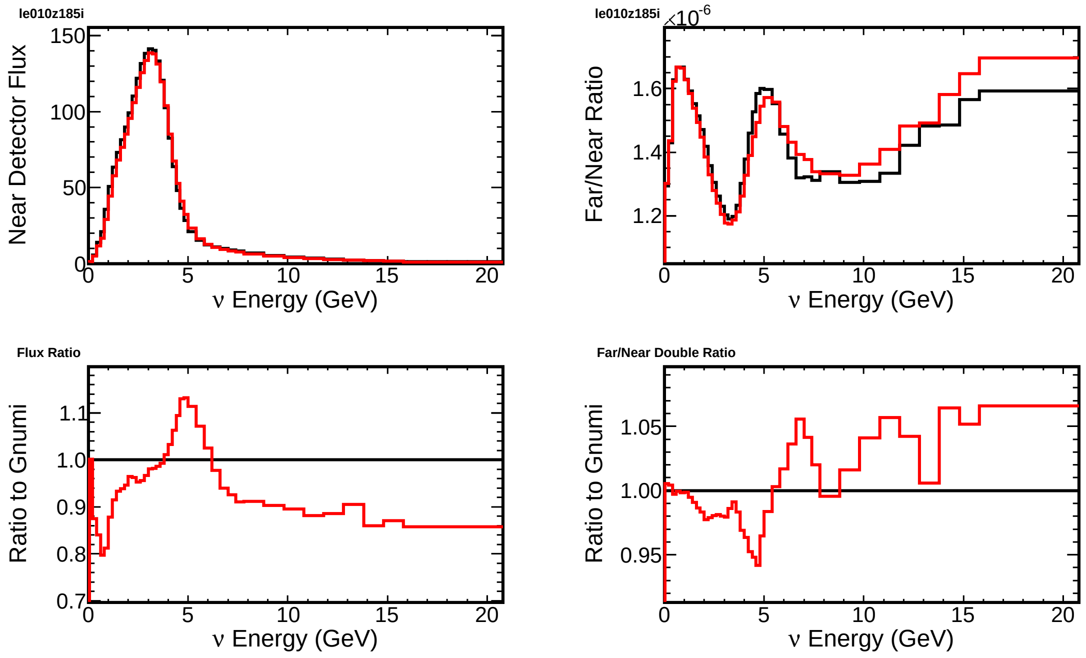

Figure 20: le010z185i, ν_μ, with helium in the decay pipe

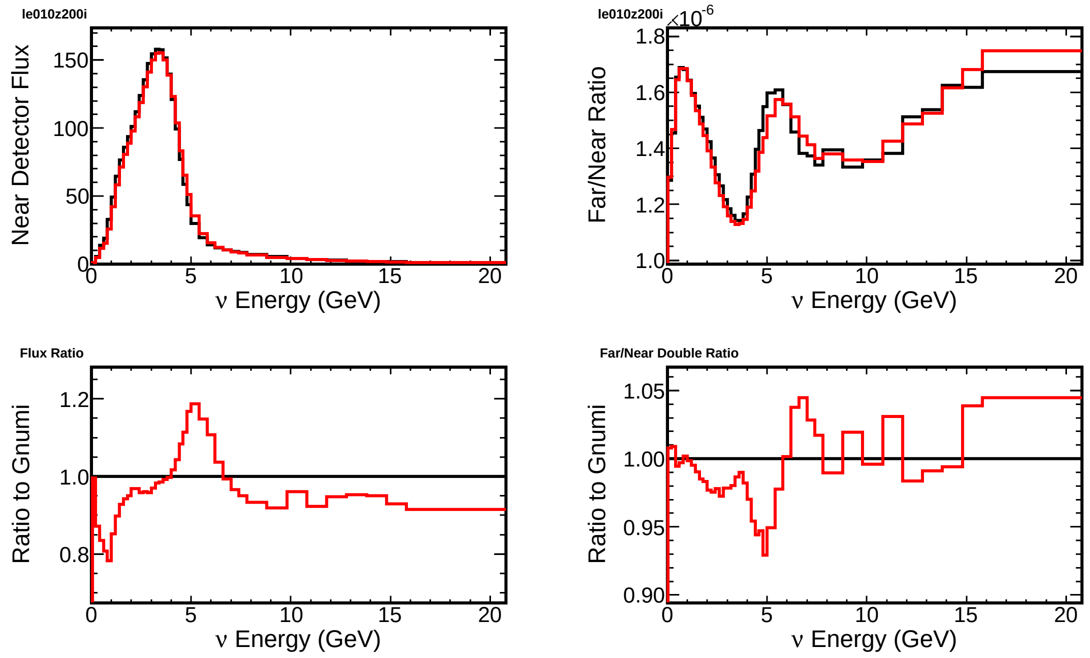

Figure 21: le010z200i, ν_μ

Figure 22: le100z200i, ν_μ

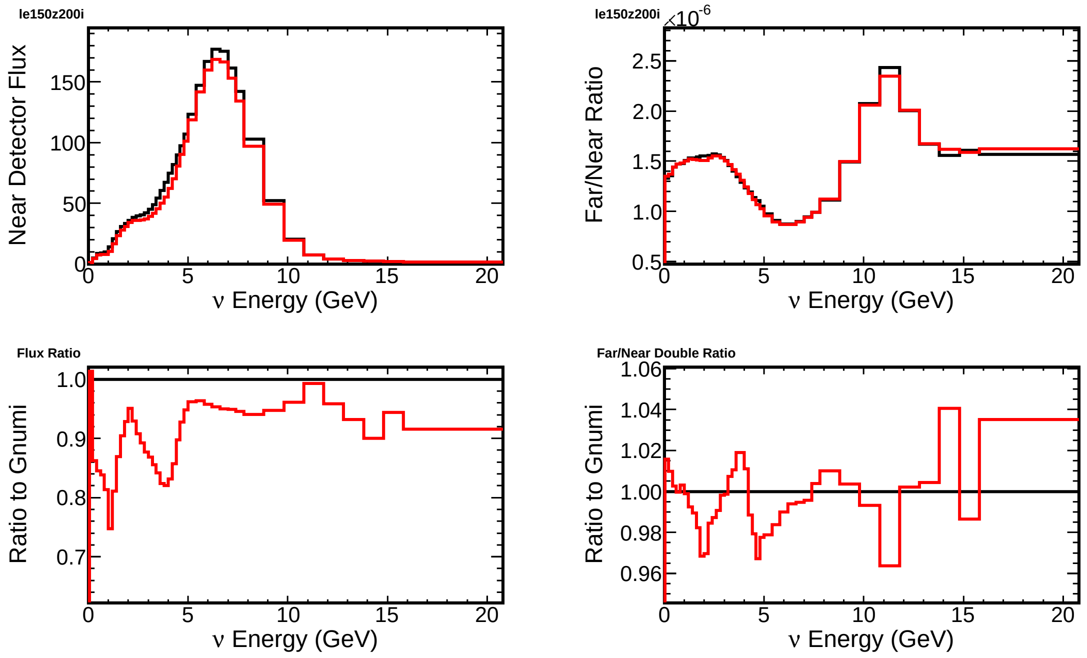

Figure 23: le150z200i, ν_μ

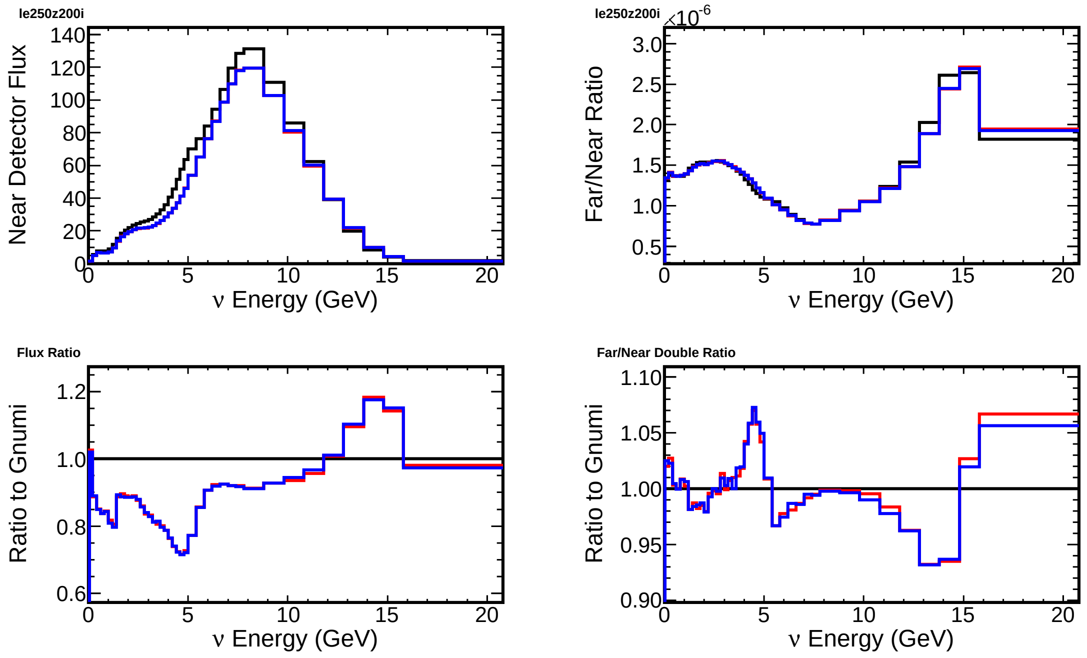

Figure 24: le250z200i, ν_μ

5.2.2 Neutrino Species

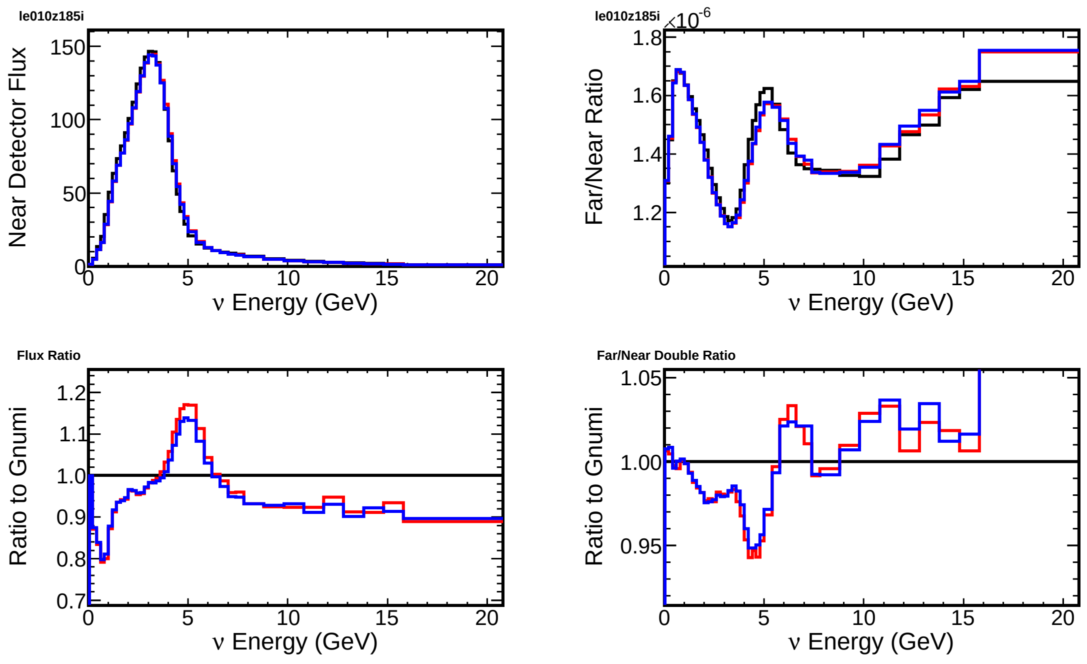

Figure 25: le010z185i, ν_μ

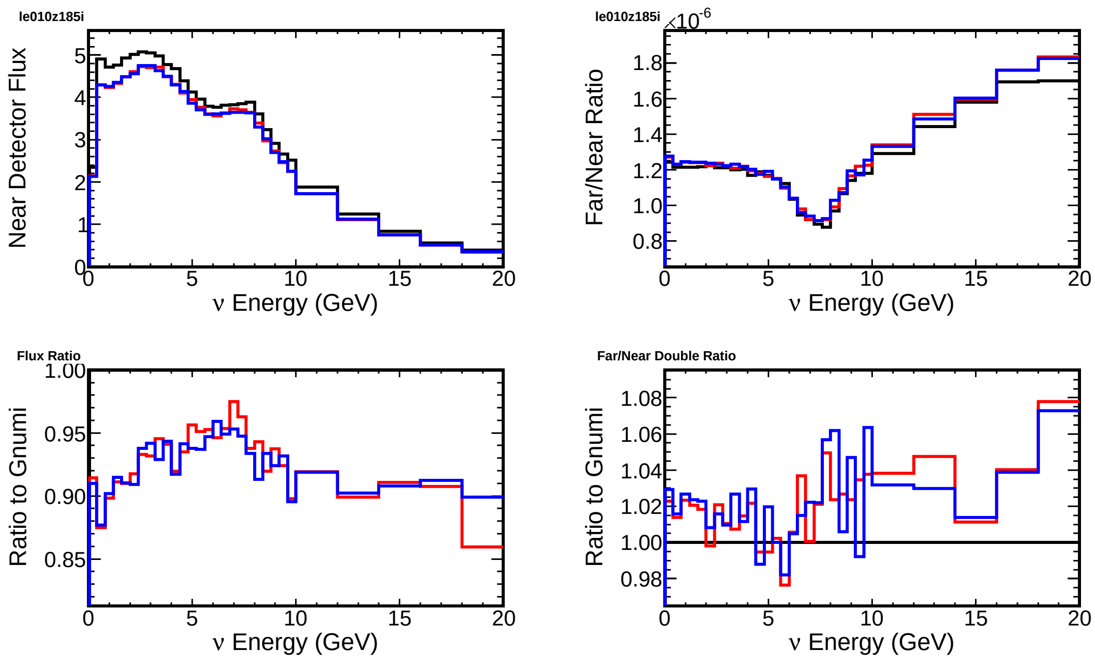

Figure 26: le010z185i, $\bar{\nu}_\mu$

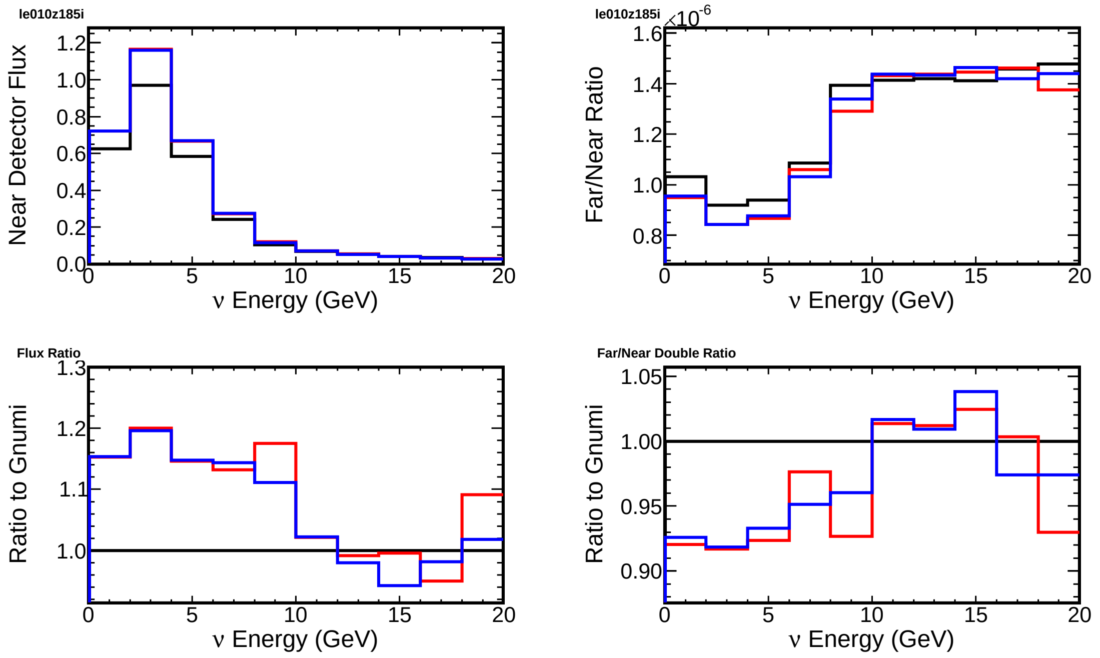

Figure 27: le010z185i, ν_e

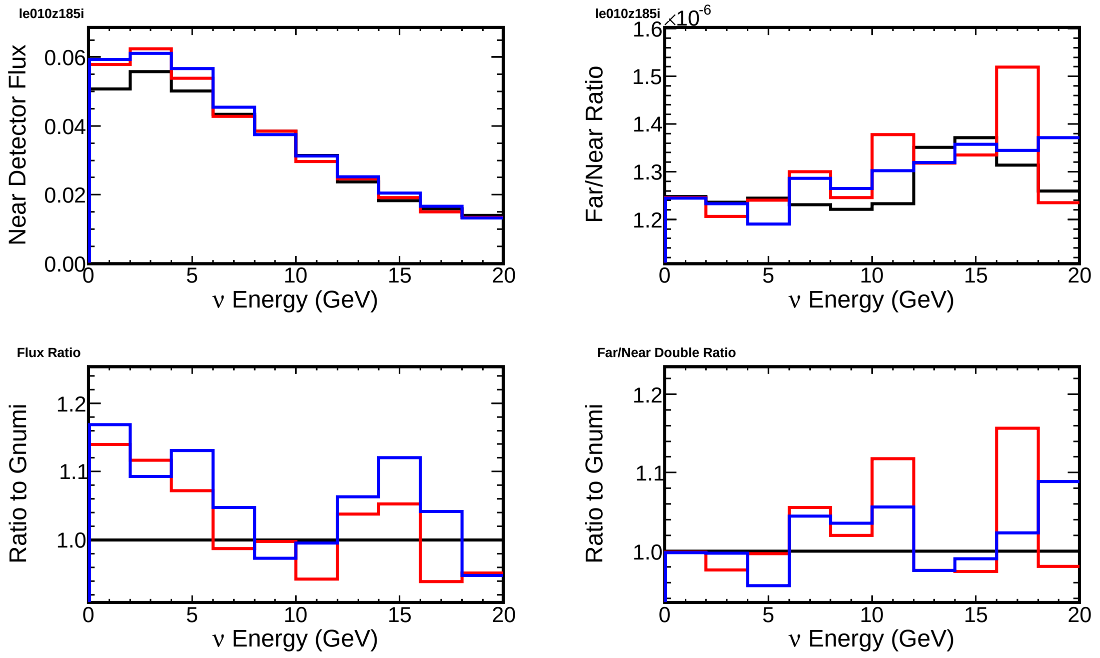

Figure 28: le010z185i, $\bar{\nu}_e$

5.2.3 Parent Species

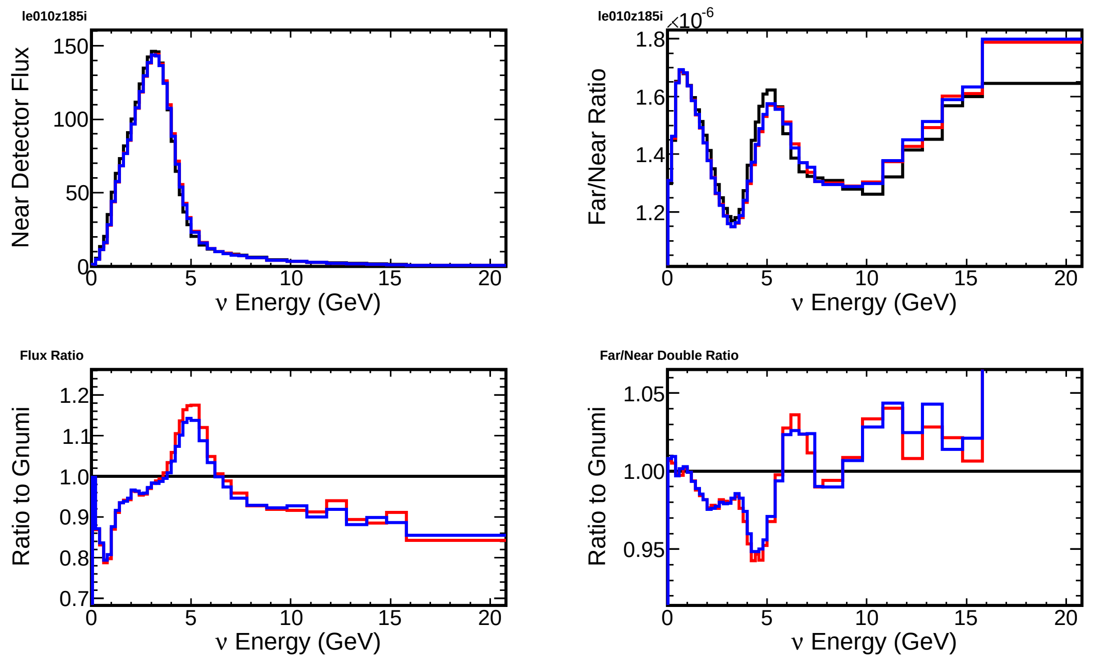

Figure 29: le010z185i, π^+

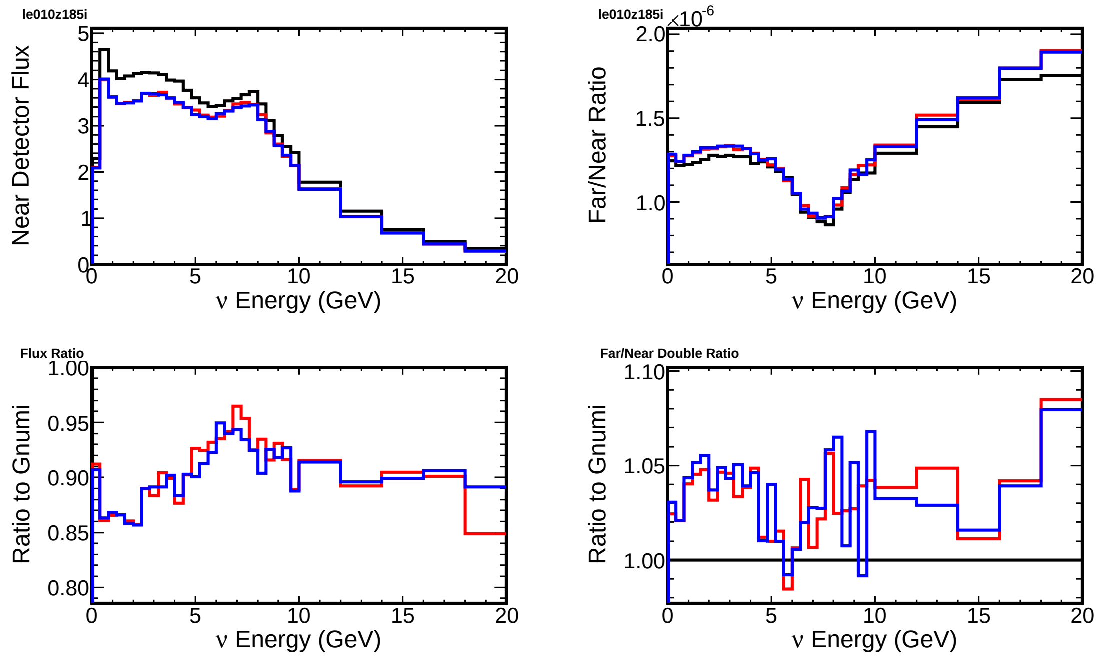

Figure 30: le010z185i, π^-

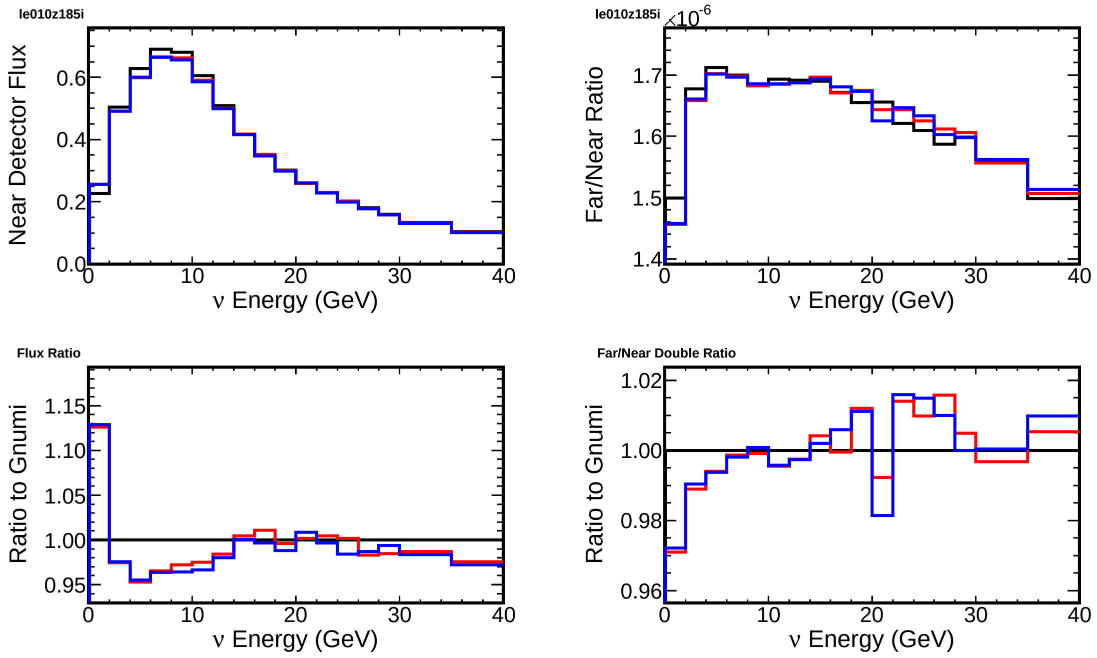

Figure 31: le010z185i, K^+

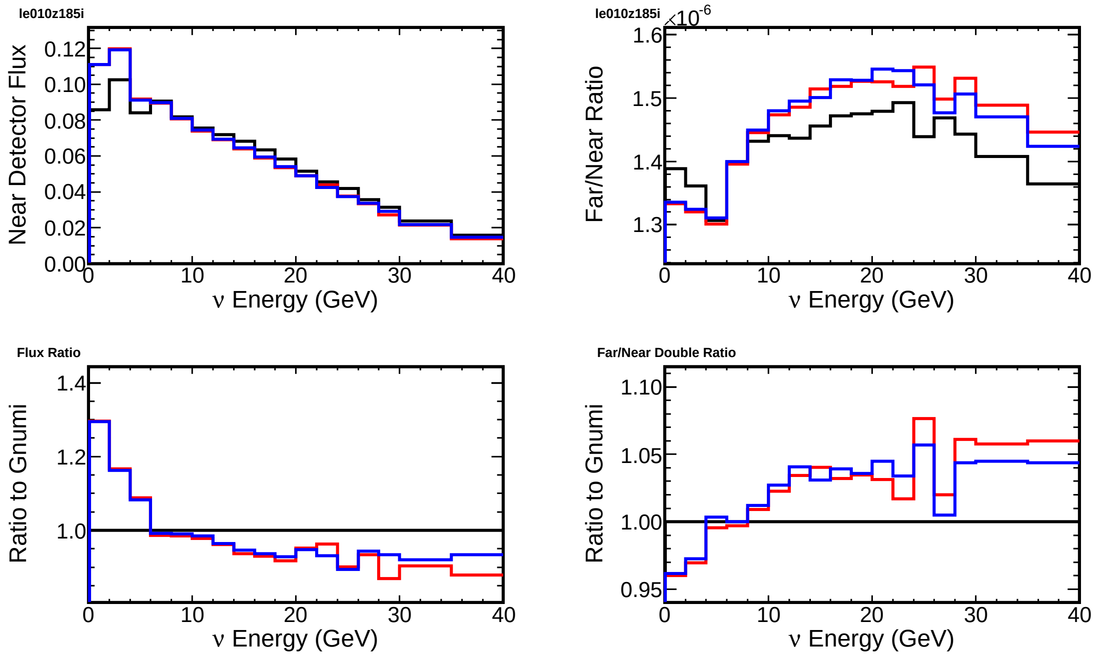

Figure 32: le010z185i, K^-

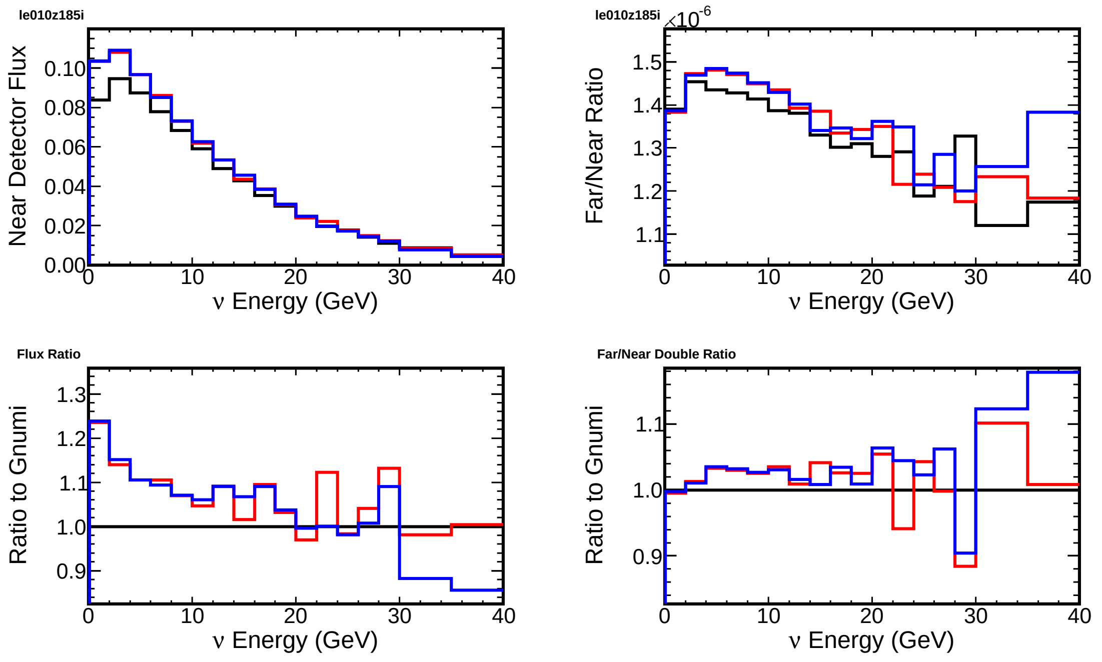

Figure 33: le010z185i, K_L^0

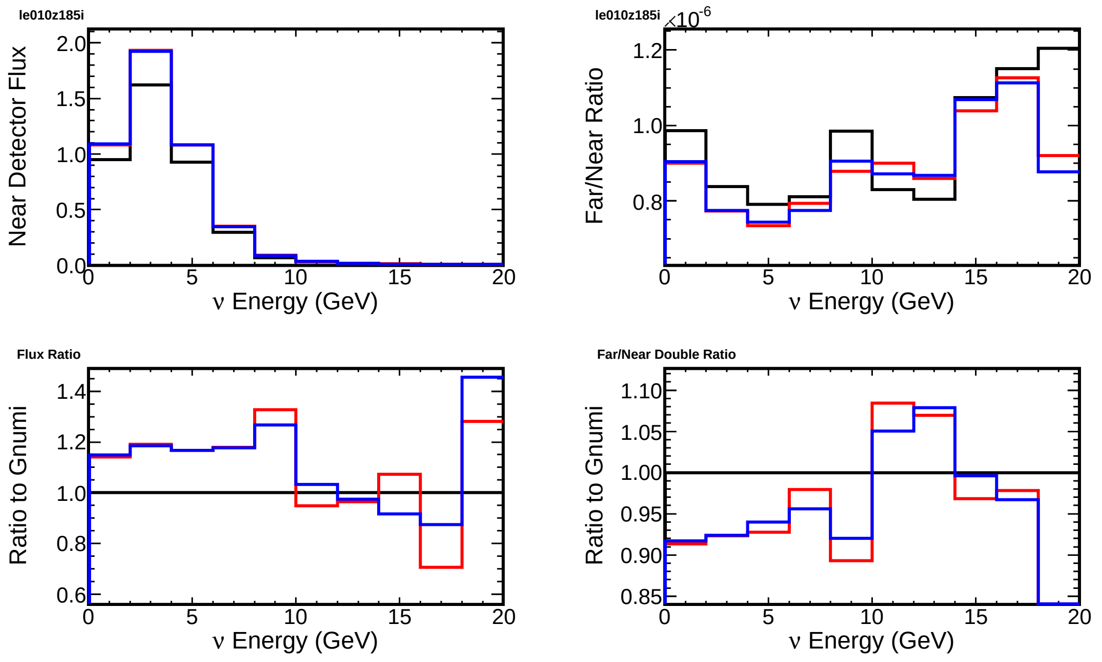

Figure 34: le010z185i, μ^+

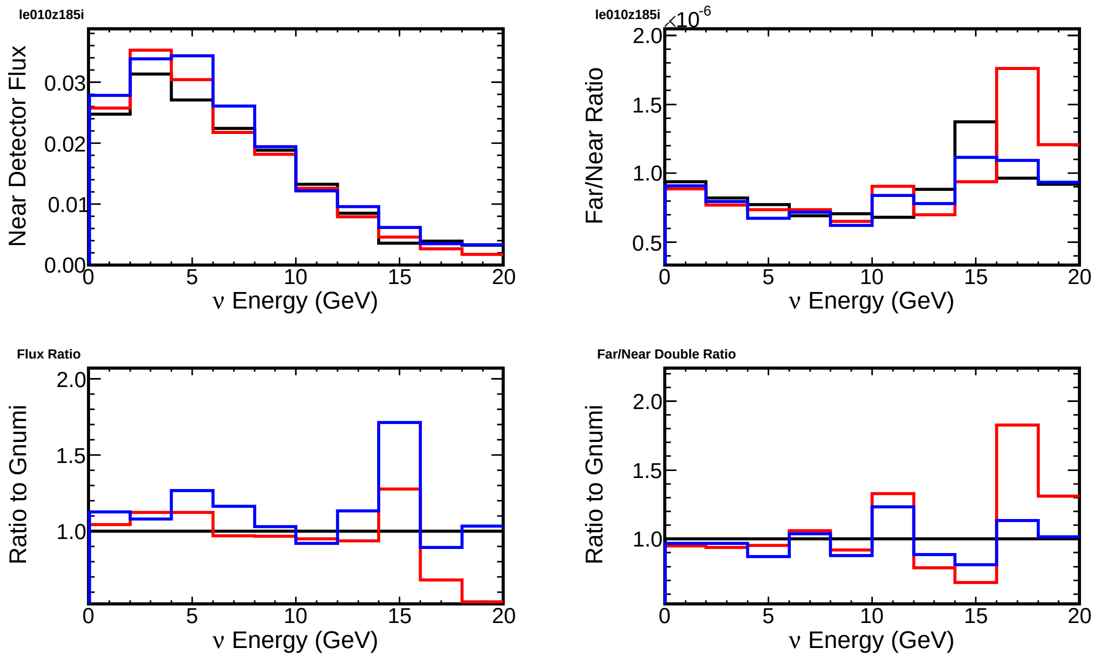

Figure 35: le010z185i, μ^-

References

[1] G. Battistoni, F. Cerutti, A. Fasso, A. Ferrari, S. Muraro, J. Ranft, S. Roesler, and P. R. Sala. The fluka code: description and benchmarking. *AIP Conference Proceedings (Hadronic Shower Simulation Workshop)*, 896:31–49, 2007.

[2] M. Campanella, A. Ferrari, P.R. Sala, and S. Vanini. Reusing code from fluka and geant4 geometry. ATL-SOFT-98-039, 13 October 1998.

[3] M. Campanella, A. Ferrari, P.R. Sala, and S. Vanini. First calorimeter simulation with the flugg prototype. ATL-SOFT-99-004, 10 December 1999.

[4] Alysia Marino. Notes on gnumi v19 flux files. MINOS-doc-2153, 25 August 2006.